Lecture 4 - Polygon Triangulation

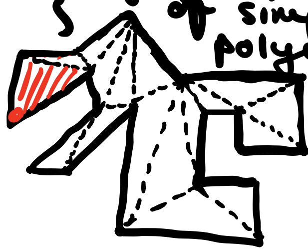
Motivation - Art Gallery Problem

Art Gallery

~ simple polygon
no holes

How many cameras does it take to guard the art gallery?

triangulation of simp. polygon



Upper bound: no. of triangles required to divide up the polygon (triangulation)

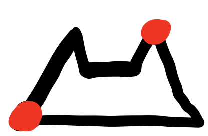
Theorem: Any simple polygon can be triangulated: if it has n vertices, we require $n-2$ triangles.

So an upper bound for the number of triangles is $n-2$.

Note: In fact, can do better - $\lfloor n/3 \rfloor$ cameras suffice

this argument uses triangulation (& 3-colouring - see book)

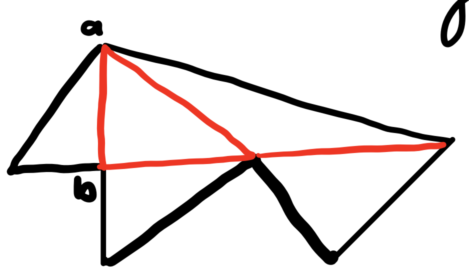
$\lfloor n/3 \rfloor$ sometimes required.



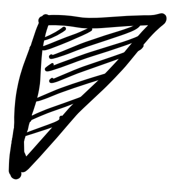
6 vertices
$\lfloor 6/3 \rfloor = 2$.

Today: polygon triangulation algorithm

Theorem: Each simple polygon can be triangulated. Moreover, any triangulation of polygon with n vertices has $n-2$ triangles.

E.g. 

7 vertices
5 triangles

Proof $n=3$  is obvious.

Will prove by induction.

- By a diagonal, we mean a straight line segment $\overrightarrow{ab}$, whose endpoints are vertices and which otherwise belongs to the interior of the polygon. (E.g. $\overrightarrow{ab}$ in example above)
- A diagonal $\overrightarrow{ab}$ splits polygon $P = A \underset{\overrightarrow{ab}}{\cup} B$ into two parts A, B with m, k vertices where

$m+k = n+2$ & $m, k < n$.

- Triangulations of A&B can be combined - so result on existence of a Triangulation follows by induction if we can prove that a diagonal always exists.

Also formula for no. of triangles follows:

$$\underset{n \text{ verts}}{P} = \underset{m}{A} \underset{ab}{\cup} \underset{k}{B}$$

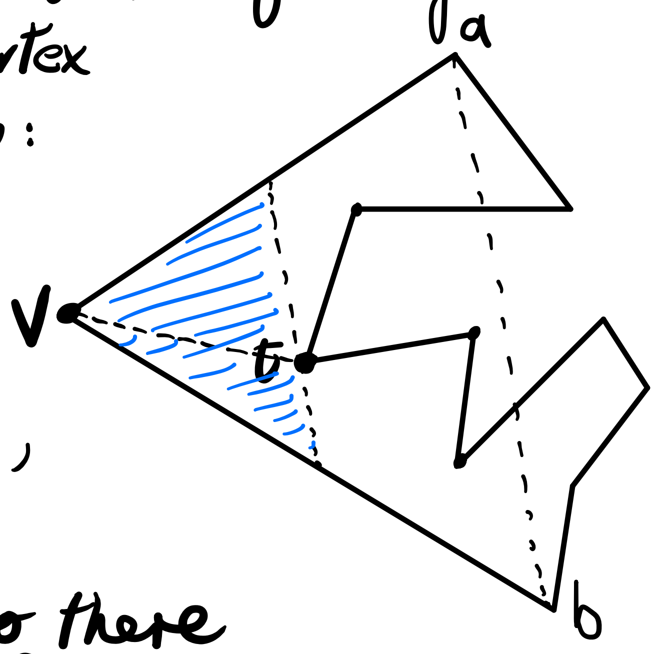
by ind. we have triang. of A in $m-2$ triangles, B ... $k-2$ triangles

So P has triang. in

$$(m-2)+(k-2) \text{ triangles}$$

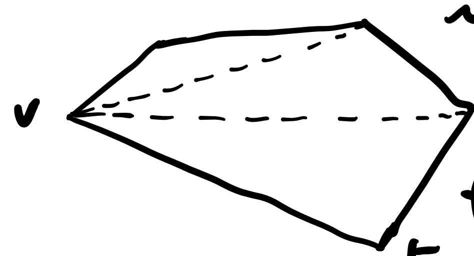
$$= m+k-4 = n-2 \text{ Triangles.}$$

- Must prove the existence of diagonal:
- Let v be smallest vertex on P lexicographically: x co-ord, then y-co-ord.
- Let a, b be vertices connected to v
- If $\overrightarrow{ab}$ is a diagonal, we are done!
- Otherwise, an edge must cross $\overrightarrow{ab}$, so there exist vertices of P lying inside $\triangle vab$
- Let t be the furthest such vertex from $\overrightarrow{ab}$
- If $\overrightarrow{vt}$ did not lie in interior, an edge would have to cross it, & one of its endpoints would lie in blue region above (ie. closer to v than t). But this is impossible.
- Therefore $\overrightarrow{vt}$ is a diagonal. $\square$



Goal: Find algorithm with complexity $O(n \log n)$.

Convex polygon:



~ easy to triangulate: draw line from vertex v to all others.

Less restrictive notion (still easily triangulable) of monotone polygon:

monotone with respect to axis y:

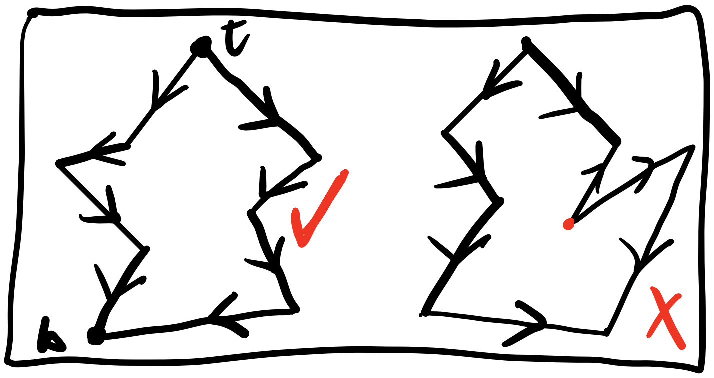
any horizontal line $y=k$ intersects polygon in line segments, point or empty set.

Today, we with slightly stronger notion of monotone polygon

Monotone polygon:

Consider lex ordering:

$a > b \iff a_y > b_y$ or $(a_y = b_y \;\&\; a_x < b_x)$



- Determines two paths from top t to bottom b.
- Polygon P is monotone if both paths are decreasing (with respect to lex. ordering)

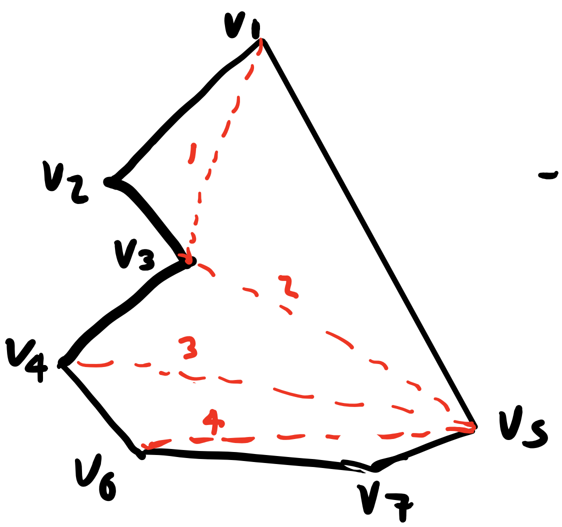
Algorithm:

1. Divide simple polygon into y-monotone pieces.
2. Triangulate monotone polygons.

This week, we do 2

Next week, do 1.

Triangulating monotone polygon



Idea: lex. ordering on vertices.
- Draw diagonals to all possible preceding vertices (lying within polygon) & break off triangles.

Formally: store monotone polygon in <u>DCEL D</u>.
Output: DCEL with diagonals added capturing triang. polygon.

Start of alg: - calculate <u>left & right paths</u> from top vertex to bottom.
- <u>Merge</u> two paths into a lex. ordered list $u_1, \ldots, u_n$.

- Initialise empty stack S.
  Push v_1 & v_2 onto it ~ so $S = (v_2, v_1)$

- At next vertex v in list, triangulate as much as possible by adding edges from v, popping the stack.
- Push v onto stack.

- At v_n, add diagonals to all but first & last.

More detailed description – did not give in class

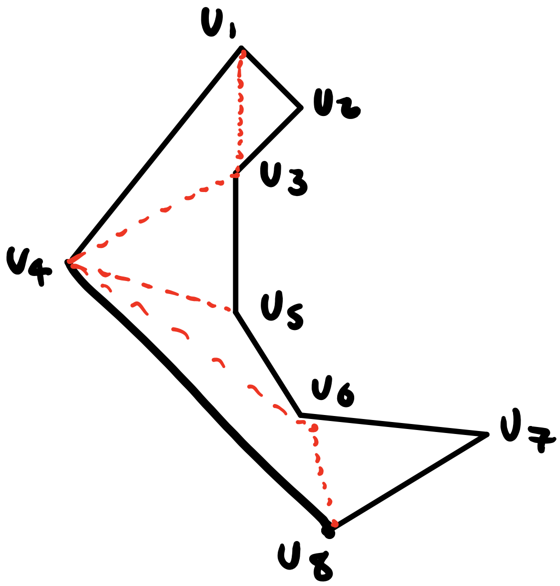
- Initialise empty stack S.
  Push v_1 & v_2 onto it ~ so $S=(v_2, v_1)$

For $j=3$ to $n-1$:

- if v_j & vertex on top of S are on diff. paths:
  - pop all vertices from S
  - add a diagonal (in D) to each popped vertex except the last one.
  - push v_{j-1} & v_j onto S.

- Otherwise,
  - pop top vertex of S.
  - pop remaining vertices as long as diagonals from v_j to them lie inside P.
  - add the diagonals to D.
  - Push last popped vertex back onto S. Push v_j onto S.

- At v_n, add diagonals to all vertices in S except the first and last ones.



- $S = (v_2, v_1)$
- At v_3, pop v_2 so $S = (v_1)$
- pop v_1 & add v_1v_3 to D.
- Push v_1 & v_3 onto S so $S = (v_3, v_1)$.

• At v_4, pop all vertices from S. Add diag v_3v_4 to D. $S = (v_4, v_3)$.

• At v_5, add diag v_4v_5 to D. $S = (v_5, v_4)$

• At v_6, pop v_5 → $S = (v_4)$. Add v_4v_6 to D.
Pop, push → Then $S = (v_6, v_4)$.

• At v_7, pop v_6 so $S = (v_4)$. Diag. $v_4v_7 \notin P$. Push → (v_7, v_6, v_4)

• At v_8, add diag. v_8v_6 to D.

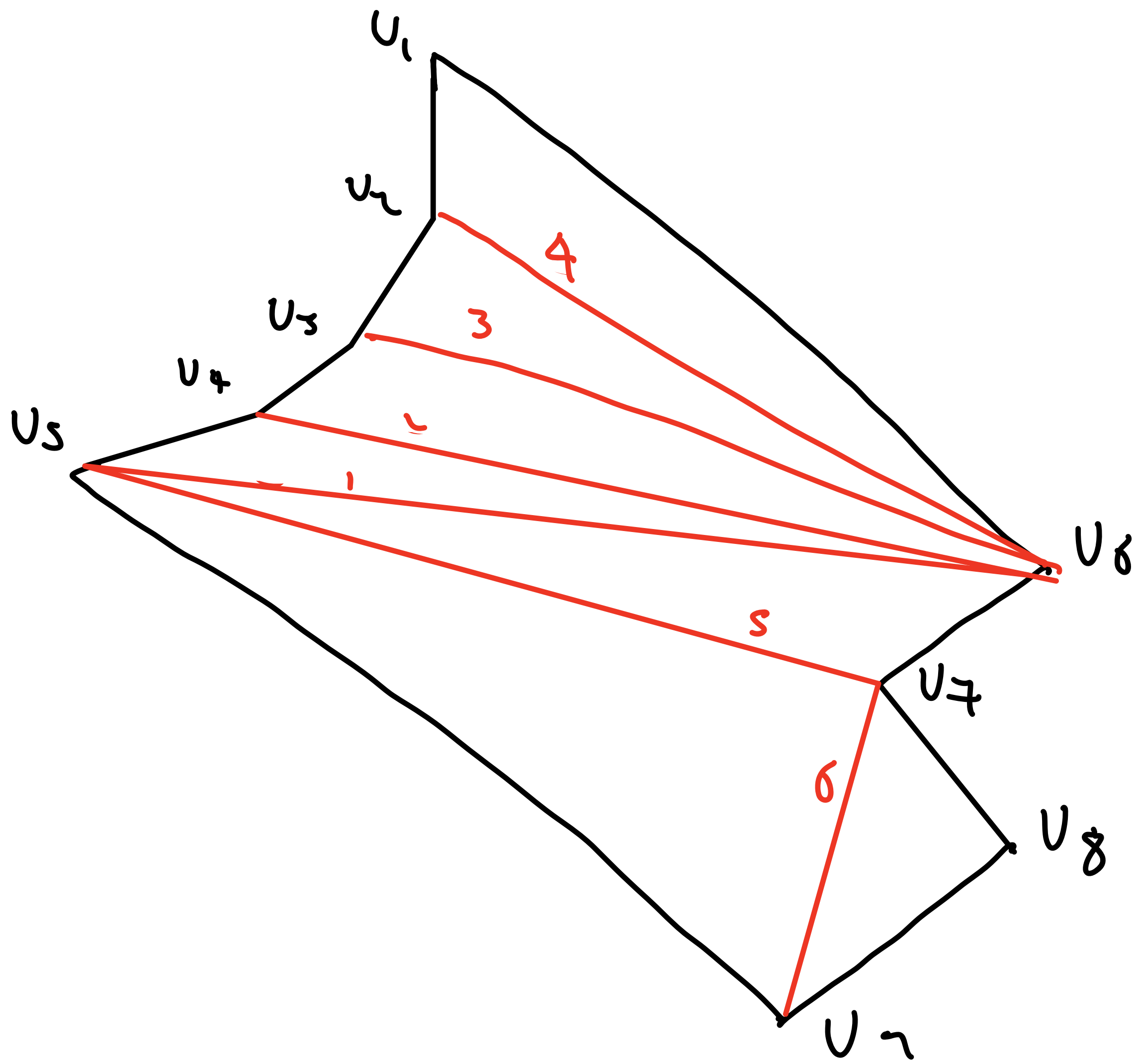
Complexity:

- calc. top, bottom $O(n)$
- calc. paths top to bottom $O(n)$
- Merge two paths takes time $O(n)$.
- During running of loop, each vertex is added / removed at most 2 times
  - so $O(n)$.
- Total: $O(n)$.

Next week,

1. Break simple polygon into monotone parts.

Combining with above, obtain alg. for triangulating a simple polygon.



v1
v2
v3
v4
v5
v6
v7
v8
v9
1
2
3
4
5
6